

The World of Combinatorial Fuzzy Problems and the Efficiency of Fuzzy Approximation Algorithms*

TOMOYUKI YAMAKAMI[†]

Abstract: We re-examine a practical aspect of combinatorial fuzzy problems of various types, including search, counting, optimization, and decision problems. We are focused only on those fuzzy problems that take series of fuzzy input objects and produce fuzzy values. To solve such problems efficiently, we design fast fuzzy algorithms, which are modeled by polynomial-time deterministic fuzzy Turing machines equipped with read-only auxiliary tapes and write-only output tapes and also modeled by polynomial-size fuzzy circuits composed of fuzzy gates. We also introduce fuzzy proof verification systems to model the fuzzification of nondeterminism. Those models help us identify four complexity classes: Fuzzy-FPA of fuzzy functions, Fuzzy-PA and Fuzzy-NPA of fuzzy decision problems, and Fuzzy-NPAO of fuzzy optimization problems. Based on a relative approximation scheme targeting fuzzy membership degree, we formulate two notions of “reducibility” in order to compare the computational complexity of two fuzzy problems. These reducibility notions make it possible to locate the most difficult fuzzy problems in Fuzzy-NPA and in Fuzzy-NPAO.

1 Background and Results

Our purpose is to (1) make a theoretical groundwork necessary to carry out practical analyses of “generalized” fuzzy problems that have naturally arisen in industrial applications and (2) lay out a theoretical framework for “generalized” fuzzy algorithms that efficiently solve those “generalized” fuzzy problems.

1.1 Practical Realm of Fuzzy Problems

In real-life situations, many objects naturally embody certain degrees of *fuzziness*, which can be in general expressed in terms of the uncertainty, ambiguity, vagueness, or imprecision of the objects. Based on a non-standard logic, Zadeh [13] tried to capture this fuzziness mathematically by formulating a basic concept of *fuzzy set*. In his theory, an ordinary object without any fuzziness is distinctly called as a “crisp” object, whereas a fuzzy object is a combination of crisp object and its membership degree (or certainty degree) that indicates intuitively how likely the crisp object actually exists. Since its introduction, fuzzy theory has found numerous applications from digital image processing to voice recognition, to telecommunication, further to a field of medicine and agriculture (see, e.g., [1, 7]). To handle a wide variety of practical fuzzy problems, we still need to lay out a groundwork in developing a general, coherent theory of fuzzy problems and fuzzy algorithms that efficiently solve them.

In the rest of this paper, we wish to limit our interest within combinatorial problems, which include search, counting, optimization, and decision problems. Let us first recall that, in an ordinary theory, those problems are viewed as functions that map every input instance to its desirable solution (or solutions). In the past literature, there have been numerous ways to fuzzificate ordinary combinatorial problems and those fuzzification methods significantly vary, depending on target areas of interests. We wish to re-examine a *fuzzification* of those problems to develop our general framework.

In many abstract treatments of fuzzy problems, input instances are merely pairs of crisp objects and their membership degrees, instead of more general “fuzzy objects.” To expand a scope of fuzzy-logic applications, it is more desirable to deal with the case where an input instance is a series of admissible fuzzy objects (or fuzzy data) and an output could be a desired “fuzzy object” as a solution derived from the given input. Formally, we define our “fuzzy problem” as a mapping from each series of fuzzy objects to another fuzzy object; thus, the term “fuzzy problem” becomes a synonym of “fuzzy functions.” To describe those fuzzy problems, we need to specify two items: *fuzzy (input) instance* and *fuzzy output*. For technical reason, we assume that the support of any fuzzy instance is finite, where the *support* of a fuzzy object over universe U is an ordinary set of elements in U having positive membership degrees. To distinguish our fuzzy problems

*This extended abstract appeared in the Proceedings of the Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS 2014) and 15th International Symposium on Advanced Intelligent Systems (ISIS 2014), December 3–6, 2014, Institute of Electrical and Electronics Engineers (IEEE), pp. 29–35, 2014.

[†]Present Affiliation: Department of Information Science, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

from the aforementioned conventional fuzzy problems, we sometimes emphasize the generosity of our fuzzy problems by calling them *generic fuzzy problems*.

(GENERIC) FUZZY PROBLEM P :

- FUZZY INSTANCE: a series (s_1, s_2, \dots, s_k) of admissible fuzzy objects having finite supports.
- FUZZY OUTPUT: an admissible fuzzy object as a solution to this input (s_1, s_2, \dots, s_k) .

Due to the page limit, we will discuss in later sections only two kinds of (generic) fuzzy problems: fuzzy decision problems and fuzzy optimization problems. *Fuzzy decision problems* are treated as a special case of the fuzzy problems whose outputs are limited to fuzzy subsets of $\{0, 1\}$. This gives rise to a new realm of fuzzy languages, each of which indicates input's "acceptance" and "rejection" with separate precision degrees, whereas conventional "fuzzy languages" are defined simply as fuzzy subsets of Σ^* for underlying alphabet Σ .

Fuzzy optimization problems are also a variant of (generic) fuzzy problems, which in general have the form $(I, SOL, m, goal)$, where I is a finite set of fuzzy (admissible) input instances, SOL is a fuzzy function listing all fuzzy solutions, m is a fuzzy measure (or objective) function from $I \circ SOL$ to natural numbers, and $goal \in \{\max, \min\}$. Here $I \circ SOL$ means the set $\{(s, t) \mid s \in I, t \in SOL(s)\}$. Let $m^*(x)$ denote the optimal value $m(x, y)$ over all solutions $y \in SOL(x)$.

1.2 Practical Realm of Fuzzy Algorithms

In Section 1.1, we have discussed our (generic) fuzzy problems. To solve those problems efficiently, we need to clarify how we model efficient fuzzy algorithms so that we can explore computational complexity issues of the fuzzy problems in a wider, generic framework.

We intend to model fuzzy algorithms by extending the existing realms of "sequential computation" and "parallel computation." Firstly, we will model our fuzzy algorithms by refining the existing notion of fuzzy Turing machines, which were considered by Zadeh [14], Lee and Zadeh [10], and Santos [12] as a fuzzification of ordinary nondeterministic Turing machines (or NTMs, in short). In their formulations, deterministic fuzzy computation inherently embodies ordinary nondeterminism, and therefore it is not surprising to know that, under a certain suitable fuzzification of ordinary instances, fuzzy algorithms solve NP-complete problems in polynomial time (cf. Proposition 6.2).

To cope with a wide range of practical fuzzy problems, Doostfatemeleh and Kremer [4] suggested how to expand the then-existing models of fuzzy algorithms by supplementing extra "safe" auxiliary operators to tune up the behaviors of those algorithms. Based on their spirit, we give a new formulation of *deterministic fuzzy Turing machines* (or DFTMs).

Another natural model that can represent fuzzy algorithms is *fuzzy (logic) circuits* (e.g., [6, 8]). In particular, we consider fuzzy circuits in which fuzzy gates are *layered* level by level. Such a fuzzy circuit takes a series of fuzzy input bits and proceeds by applying fuzzy gates at each level. This mechanism can realize fuzzified "parallel computation." We prove that families of polynomial-size fuzzy circuits are equivalent in computational power to polynomial-time DFTMs.

The minimum amount of computational resources necessary to solve given problems is of great concern from a practical viewpoint. As in ordinary computational complexity theory, we are also interested in fuzzy algorithms running in polynomial time. For the purpose of solving a wider scope of fuzzy problems, we look for an approximation of the outcomes of fuzzy algorithms. We therefore define Fuzzy-FPA(γ) as the class of fuzzy functions (or equivalently, fuzzy problems having output values) computed approximately with relative closeness of γ by DFTMs in polynomial time. Restricted to fuzzy decision problems, we also obtain Fuzzy-PA(γ) as a natural fuzzification of the complexity class P.

Nondeterminism has been modeled as a proof-verification process, in which, for positive instances, there exists a "proof" (i.e., necessary key information) for which a "verifier" can easily confirm its validity and, for negative instances, the verifier refutes any proof provided to him as invalid. We recognize a naturally-induced class Fuzzy-NPA(γ) of fuzzy decision problems that are solved approximately with relative closeness γ by such fuzzy proof verification systems (or FPVS's, in short) in polynomial-time.

To discuss the computational complexity of fuzzy problems, we introduce a fundamental notion of *polynomial-time approximate fuzzy reducibility* (abbreviated as AF-reducibility) between two fuzzy problems. Such a reducibility notion among fuzzy decision problems makes it possible to designate "complete" problems, which indicate the most difficult problems in a given class of problems to solve in polynomial time, as for the notion of NP-complete problems (see [5]). We prove the existence of complete problems in Fuzzy-NPA(1) under AF-reductions.

For fuzzy optimization problems, similarly to Fuzzy-NPA(γ), we define Fuzzy-NPAO(γ), which is composed of all fuzzy optimization problems characterized by fuzzy functions in Fuzzy-FPA(γ). Between two fuzzy optimization problems, we also introduce a notion of *polynomial-time approximation-preserving fuzzy reducibility* (or APF-reducibility) and prove that Fuzzy-NPAO(1) contains complete problems under APF-reductions.

All omitted or abridged proofs will appear in a complete version of this extended abstract.

2 Basic Notions and Notations

The notation \mathbb{Z} (resp., \mathbb{Q} , \mathbb{R}) denotes the set of all integers (resp., rational numbers, real numbers). We use \mathbb{N} to denote the set of all natural numbers (i.e., non-negative integers) and we set $\mathbb{N}^+ = \mathbb{N} - \{0\}$. For two integers n, m with $n \leq m$, $[n, m]_{\mathbb{Z}}$ stands for the *integer interval* $\{n, n+1, n+2, \dots, m\}$. For simplicity, we write $[n]$ to denote the interval $[1, n]_{\mathbb{Z}}$ whenever $n \geq 1$. The notation $[0, 1]^*$ (resp., $[0, 1]^+$) denotes the union of $[0, 1]^k$ for any constant $k \in \mathbb{N}$ (resp., $k \in \mathbb{N}^+$). Given any propositional formula F , we set $[F]$ to be 1 if F is true, and 0 otherwise. For example, $[x = y]$ equals 1 if and only if $x = y$. For any set A , $\mathcal{P}(A)$ denotes the *power set* of A .

An *alphabet* Σ is a finite nonempty set of “symbols” and a *string* over Σ is a finite sequence of symbols taken from Σ ; in particular, the *empty string* is always denoted λ . Let Σ^* denote the set of all strings over Σ . For any logical proposition (or logical statement) P , the notation $[P]$ denotes 1 if P is true, and 0 otherwise. For example, we have $[x = y] = 1$ if x equals y , and $[x = y] = 0$ if x is different from y .

Due to the page limit, we assume that the reader is familiar with basic concepts of fuzzy sets. Although a *fuzzy (sub)set* A of universe U is a map from U to $[0, 1]$, it is also viewed as an ordinary set that is composed of pairs of the form (x, γ) , where $x \in U$ and $\gamma \in [0, 1]$; that is, $(x, \gamma) \in A$ iff $\gamma = A(x)$. To simplify the descriptions of fuzzy sets in the rest of the paper, we will take those two different viewpoints interchangeably. For convenience, we write $\mathcal{F}(U)$ for the collection of all fuzzy subsets of U . Moreover, we use the notations $\text{core}(s)$ for the *core* of fuzzy set s and $\text{Supp}(s)$ for the *support* of s .

Definition 2.1 (fuzzy function) Given two universes U_1 and U_2 , let $A \subseteq \mathcal{F}(U_1)$ and $B \subseteq \mathcal{F}(U_2)$. A (*generic*) *fuzzy function* f from A to B , written as $f : A \rightarrow B$, satisfies that, for every $s \in A$, $f(s)$ is a fuzzy subset of U_2 in B .

Remark: It may be possible to expand our definition of fuzzy sets by replacing the unit interval $[0, 1]$ with an arbitrary complete lattice and expand our results further. However, we leave such a challenging task to the interested reader.

3 Combinatorial Fuzzy Problems

In pursuit of simplicity, we limit our attention within “combinatorial problems” that inherently embody fuzziness. Now, we want to explain what “fuzzy problems” are. As described in Section 1.1, fuzzy problems with which we deal in this paper take series of fuzzy instances (such as fuzzy graphs, fuzzy numbers, fuzzy functions, etc.) as inputs and produce certain fuzzy objects as outputs. We treat such a problem as a mapping from a set of fuzzy objects (or fuzzy instances) to another fuzzy object.

As customarily in ordinary computational complexity theory, we will deal only with *discrete* objects as our input instances. For a general treatment of such discrete fuzzy instances, it is useful to assume a suitable encoding of those instances to a certain fixed object. In a way similar to defining “strings over alphabet” in ordinary language theory, we will introduce an exquisite notion of “fuzzy string,” which expresses a certain degree of imprecision, uncertainty, or incompleteness of string information. Our definition of fuzzy string is inspired by the notion of *LR-shape fuzzy numbers* (see, e.g., [7]) and is closely related to “discretized” fuzzy numbers.

A *fuzzy string quantity* s over Σ is an element of $\mathcal{F}(\Sigma^*)$, namely, a fuzzy subset of Σ^* . We call each value $s(x)$ a *precision degree* of x instead of “membership degree.” A *fuzzy string* over Σ (with respect to a suitably chosen distance measure d , e.g., the Hamming distance between two crisp strings) is a fuzzy string quantity s over Σ that satisfies the following condition: there exist a crisp string $x_0 \in \Sigma^*$ and a crisp function $\eta : [0, 1] \rightarrow [0, 1]$ (which may depend on x_0) such that (1) $\text{Supp}(s)$ is a finite set, (2) η is strictly decreasing function with $\eta(0) = 1$ and $\eta(1) = 0$, and (3) $\text{Cut}_{\gamma}(s) \subseteq \text{Ball}_d(x_0, \eta(\gamma))$ holds for any real number $\gamma \in [0, 1]$, where $\text{Ball}_d(x, \gamma) = \{z \in \Sigma^* \mid d(x, z) \leq \gamma\}$. This special string x_0 is called a *target* of the fuzzy string s .

Let $F\Sigma^*$ denote a collection of all (possible) fuzzy strings over Σ . It holds that $\text{core}(s) = x_0$ by Conditions (2) and (4). Note that a fuzzy string is a convex fuzzy set. As usual in fuzzy theory, a fuzzy string s can be seen as the set $\{(x, s(x)) \mid x \in \Sigma^*\}$. We also follow a standard convention that, when a fuzzy string s is crisp with a target x_0 (i.e., $s(x_0) = 1$ and $s(x) = 0$ for any $x \in \Sigma^* - \{x_0\}$), we identify s with x_0 . This helps us treat all crisp strings as a special case of fuzzy strings.

Henceforth, we will focus our attention only on any subset $F\Theta$ of $F\Sigma^*$ for a certain alphabet Σ and a certain distance measure d . A (*generic*) *fuzzy problem* stated in Section 1.1 is now rephrased as follows. Fix our alphabet Σ_1, Σ_2 and consider two sets $F\Theta_1 \subseteq F\Sigma_1^*$ and $F\Theta_2 \subseteq F\Sigma_2^*$.

(GENERIC) FUZZY PROBLEM L w.r.t. $(F\Theta_1, F\Theta_2)$:

- FUZZY INSTANCE: a series (s_1, s_2, \dots, s_k) of fuzzy strings in $F\Theta_1$.
- FUZZY OUTPUT: a fuzzy string in $F\Theta_2$.

In particular, we will study two types of fuzzy problems: fuzzy decision problems and fuzzy optimization problems, defined in Section 1.1.

4 Models for Efficient Fuzzy Algorithms

To solve the fuzzy problems described in Section 3, we want to design efficient fuzzy algorithms.

4.1 Auxiliary Operators for Practical Algorithms

The existing formulation of fuzzy Turing machine seems too restrictive to apply to many real-life fuzzy problems. For a wider range of practical applications of fuzzy machines, Doostfatemeh and Kremer [4] proposed a rigorous use of “auxiliary functions” to a design of fuzzy algorithms (actually, fuzzy finite automata in their case). For the purpose of intended practical applications, we adopt their idea and introduce four auxiliary operators $(\mu_1, \mu_2, \mu_3, \xi)$, where $\mu_1 : [0, 1]^2 \rightarrow [0, 1]$, $\mu_2 : [0, 1]^+ \rightarrow [0, 1]$, $\mu_3 : [0, 1]^+ \rightarrow [0, 1]$, and $\xi : [0, 1]^* \rightarrow [0, 1]$. The use of them will be clarified in Section 4.2.

A tuple $(\mu_1, \mu_2, \mu_3, \xi)$ of auxiliary operators is said to be *safe* if those functions satisfy the following conditions: for any finite nonempty ordered set A and any finite ordered set B ,

- (1) $\mu_1(\alpha, \alpha) = \alpha$ for any $\alpha \in [0, 1]$,
- (2) $\mu_2(\{\alpha_r\}_{r \in A}) = \alpha$ if $\alpha_r = \alpha$ for all $r \in A$,
- (3) $\mu_3(\{\alpha_r\}_{r \in A}) = \alpha$ if $\alpha_r = \alpha$ for all $r \in A$, and
- (4) $\xi(\emptyset) = 0$ and $\xi(\{\alpha_r\}_{r \in B}) = \alpha$ if $\alpha_r = \alpha$ for $r \in B$.

As a concrete example, we present a *standard* safe tuple $\Xi = (\mu_1, \mu_2, \mu_3, \xi)$, which we will use later. Let \wedge and \vee be any t-norms. We set $\mu_1(\alpha, \beta) = \alpha \wedge \beta$. For any finite nonempty ordered set A , let $\mu_2(\{\alpha_r\}_{r \in A}) = \bigvee_{r \in A} \alpha_r$ and $\mu_3(\{\alpha_r\}_{r \in A}) = \bigvee_{r \in A} \alpha_r$. Moreover, for any finite ordered set B , let $\xi(\emptyset) = 0$ and $\xi(\{\alpha_r\}_{r \in B}) = \bigvee_{r \in B} \alpha_r$.

4.2 Deterministic Fuzzy Turing Machines

In order to describe a fuzzified “sequential computation” within our framework, we use a mathematical model of *deterministic fuzzy Turing machines equipped with read-only auxiliary-input tapes, rewritable input/work tapes, and write-only output tapes* supported by safe tuples of auxiliary operators. This model is considered as an extension of ordinary nondeterministic Turing machines.

Let Δ and Γ be an *input alphabet* and an *output alphabet* including a designated blank symbol $\#$. To operate a machine, we need two more tape alphabets Σ_1 (with $\$, \# \in \Sigma_1$) and $\Sigma_2 (= \{\$, \#, 1, \#\})$ for the machine’s internal use. We fix $F\Theta_1 \subseteq \mathcal{F}(\Delta^*)$ and $F\Theta_2 \subseteq \mathcal{F}(\Gamma^*)$ and also fix a safe tuple $\Xi = (\mu_1, \mu_2, \mu_3, \xi)$ of auxiliary operators. A *deterministic fuzzy Turing machine with a write-only output tape* (abbreviated as DFTM) is a triplet $\langle M, \Xi, I \rangle$, where M is of the form $(Q, \Delta, \Sigma_1, \Sigma_2, \Gamma, \delta, q_0, F)$ and I is a target set pair $(F\Theta_1, F\Theta_2)$. Moreover, Q is a finite nonempty set of the machine’s (inner) states and δ is a *fuzzy transition function* mapping from $(Q - F) \times \Sigma_1 \times \Sigma_2$ to $\mathcal{F}(Q \times \Sigma_1 \times \Gamma \times D_1 \times D_2)$ with $D_i = \{0, \pm 1\}$, $q_0 (\in Q)$ is the initial state, and $F (\subseteq Q)$ is a set of *halting states*. Whenever M enters a halting state, since δ is no longer applicable, M is considered to *halt*.

Since Ξ and I are fixed, we often refer to M as a DFTM as long as it is clear from the context. Graphically speaking, M works as follows. A *fuzzy input instance* s to M is a fuzzy object in $F\Theta_1$. Let $\ell(s) = \max_{x \in \text{Supp}(s)} \{|x|\}$ be the *length* of s . The DFTM M takes s and eventually produces another fuzzy subset $M(s)$ of $F\Theta_2$ as its output. In an initial setup process, M receives s , it automatically generates *in*

parallel all crisp strings $x \in \text{Supp}(s)$, and writes down each x , surrounded by two endmarkers $\text{\textcircled{e}}$ and $\text{\textcircled{s}}$, on the input/work tape and the other tape cells initially hold the blank symbol $\#$. Moreover, M generates $\text{\textcircled{e}}1^{\ell(s)}\text{\textcircled{s}}$ on the read-only auxiliary input tape. All tape cells are indexed by integers and all tape heads are initially positioned at the 0th cell (where $\text{\textcircled{e}}$ must be written on the first two tapes). An inner state of M is q_0 .

Assume that, at an arbitrary moment, the tape head of M scans symbol σ_1 on the input/work tape and σ_2 on the read-only input tape in inner state q and the third tape head is situated at the first (i.e., the leftmost) blank cell. When δ is applied, M overwrites σ_1 with τ , enters state p , move the first two tape heads in directions d_1 and d_2 , and writes symbol η on the output tape, together with *possibility degree* $\delta(q, \sigma_1, \sigma_2)(p, \tau, \eta, d_1, d_2)$. If $\eta \neq \lambda$, then the output tape head must move to the right; otherwise, it stays still.

Let us explain how a DFTM generates a “fuzzy computation,” composed of a set of fuzzy computation paths (i.e., series of configurations). A *(global) configuration* is a string of the form $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ with $u, v \in \Sigma_1^*$, $q \in Q$, $r \in [0, \ell(s)]_{\mathbb{Z}}$, and $w \in \Gamma^*$, where $\text{\textcircled{e}}$ is a designated separator, and $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ indicates that an input/work tape content is uv , the second tape head scans cell r , an output tape content is w , an inner state is q , and the first and third tape heads are respectively scanning the leftmost symbol of v and the first blank symbol to the right of w . Let $CONF_M$ denote the set of all (possible) configurations of M , namely, $\{uqv\text{\textcircled{e}}r\text{\textcircled{e}}w \mid u, v \in \Sigma_1^*, q \in Q, r \in [0, \ell(s)]_{\mathbb{Z}}, w \in \Gamma^*\}$. A *final configuration* is a configuration of the form $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ with $q \in F$ in $CONF_M$. Each element of $\mathcal{F}(CONF_M)$ is called a *fuzzy instance description* or a *fuzzy configuration*.

Let us describe how a “time evolution” of fuzzy configurations takes place. To express a fuzzy configuration at time i , we inductively introduce a fuzzy set $conf_i$ ($i \in \mathbb{N}$) as follows. Let $conf_0$ be a *fuzzy configuration at time 0* defined as

$$conf_0(uqv\text{\textcircled{e}}r\text{\textcircled{e}}w) = \sum_{x \in \text{Supp}(s)} s(x) \cdot [upv\text{\textcircled{e}}r\text{\textcircled{e}}w = q_0\text{\textcircled{e}}x\text{\textcircled{s}}0\text{\textcircled{e}}\lambda]$$

for any configuration $upv\text{\textcircled{e}}r\text{\textcircled{e}}w$ of M . Next, $conf_{i+1}$ is a *fuzzy configuration at time $i+1$* whose possibility degree $conf_{i+1}(u'pv'\text{\textcircled{e}}r\text{\textcircled{e}}w')$ is defined as follows. Taking pre-determined orderings on $(Q - F) \times \Sigma_1 \times \Sigma_2$ and $D_1 \times D_2$ and also assuming that $u'pv'\text{\textcircled{e}}r\text{\textcircled{e}}w'$ has the form $u\tau_1p\tau_2\tau_3v\text{\textcircled{e}}r\text{\textcircled{e}}w\eta$ with $p \in Q - F$ and , let $conf_{i+1}(u'pv'\text{\textcircled{e}}r\text{\textcircled{e}}w')$ be

$$\mu_2(\{\ell_{d_1, d_2}(q, \sigma_1, \sigma_2)\}_{(d_1, d_2) \in D_1 \times D_2, (q, \sigma_1, \sigma_2) \in (Q - F) \times \Sigma_1 \times \Sigma_2}),$$

where $\ell_{d_1, d_2}(q, \sigma_1, \sigma_2)$ represents the following values. For example, when $(d_1, d_2) = (+1, -1)$, $\ell_{+1, -1}(q, \sigma_1, \sigma_2)$ equals

$$\mu_1(conf_i(uq\sigma\tau_2\tau_3v\text{\textcircled{e}}(r+1)\text{\textcircled{e}}w), \delta(q, \sigma_1, \sigma_2)(p, \tau_1, \eta, +1, -1)),$$

where σ_2 is the symbol at cell $r+1$. Notice that each value $\ell_{d_1, d_2}(q, \sigma_1, \sigma_2)$ is always defined. If either $p \in F$ or $u'pv'\text{\textcircled{e}}r\text{\textcircled{e}}w'$ has a wrong form, then we set $conf_{i+1}(u'pv'\text{\textcircled{e}}r\text{\textcircled{e}}w') = 0$.

A *fuzzy computation* of M on fuzzy input s is a series $(conf_0, conf_1, \dots)$ of fuzzy configurations defined above. Let $final_M$ be an element of $\mathcal{F}(\mathbb{N} \times CONF_M)$ defined as

$$final_M(t, uqv\text{\textcircled{e}}r\text{\textcircled{e}}w) = [q \in F] \cdot \mu_3(\{conf_i(uqv\text{\textcircled{e}}r\text{\textcircled{e}}w)\}_{i \in [0, t]_{\mathbb{Z}}})$$

for any $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ in $CONF_M$. This value $final_M(t, uqv\text{\textcircled{e}}r\text{\textcircled{e}}w)$ is called the *possibility degree* of the (final) configuration $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ at time t . Let $FIN_M(t, s)$ be the set of all pairs (uqv, r) with $q \in F$, $u, v \in \Sigma^*$, $|uv| \leq t$, and $r \in [0, \ell(s)]_{\mathbb{Z}}$. Obviously, $FIN_M(t, s)$ is a finite set.

Since we are interested in time-bounded computation, we define the *(absolute) running time* of M on input s to be a unique number $t \in \mathbb{N}$ for which (i) there exists a final configuration $u'q'v'\text{\textcircled{e}}r\text{\textcircled{e}}w'$ satisfying $final_M(t, u'q'v'\text{\textcircled{e}}r\text{\textcircled{e}}w') > 0$ and (ii) $final_M(i, uqv\text{\textcircled{e}}r\text{\textcircled{e}}w) = 0$ holds for all configurations $uqv\text{\textcircled{e}}r\text{\textcircled{e}}w$ and all $i > t$. For convenience, we say that M *(absolutely) halts on input s in time t* if t is at least the running time of M on s .

Now, let t be the running time of M on s . If b is a fuzzy subset of Γ^* satisfying

$$b(w) = \xi(\{final_M(t, uqv\text{\textcircled{e}}r\text{\textcircled{e}}w)\}_{(uqv, r) \in FIN_M(t, s)})$$

for every $w \in \Gamma^*$, then we say that M *outputs b on input s in time t* . For convenience, we write $M(s)$ to denote this particular fuzzy object b . Obviously, if M outputs b in time t , then $|\text{Supp}(b)| \leq |\Gamma|^t$ holds, and thus b has a finite support.

We say that a fuzzy problem L *admits* a DFTM M (or M *solves* L) if $M(s) = L(s)$ holds for any $s \in F\Theta_1$. In this case, the problem L is succinctly denoted by $\mathcal{L}(M)$.

4.3 Fuzzy Gates and Fuzzy Circuits

In an ordinary setting, uniform families of *Boolean circuits* composed of *Boolean gates* have been used to model efficient algorithms. Likewise, we will consider a fuzzification of those circuits to model our intended fuzzy algorithms. In what follows, we fix a safe tuple $\Xi' = (\mu_1, \mu_2, \mu_3, \xi)$ of auxiliary operators. A fuzzy gate is specified by a finite set of input variables attached with distinguished labels and another finite set of output variables having specific labels. A *fuzzy gate with m inputs and k outputs* simply represents a function mapping $\{0, 1\}^m$ to $\mathcal{F}(\{0, 1\}^k)$. For a physical implementation of fuzzy circuits, it may be desirable to restrict the usable types of fuzzy gates but, meanwhile, we allow any types of fuzzy gates.

In our fuzzy circuit model, a finite number of fuzzy gates are layered by “levels.” At each middle level, no two fuzzy gates must share output variables with the same label; however, fuzzy gates are allowed to share input variables with the same labels. Nevertheless, all input variables at level 0 have distinct labels. We also permit the use of “fuzzy constants” (i.e., elements in $\mathcal{F}(\{0, 1\}^\ell)$ for a fixed ℓ) as part of inputs.

Let us explain how a fuzzy circuit operates on a given set of inputs. Let (x_1, x_2, \dots, x_n) be a series of all input variables used for a given fuzzy circuit. We define a *global configuration* $conf_t$ at level t as follows. At level 0, let $conf_0(x_1 x_2 \dots x_n) = s(x_1 x_2 \dots x_n)$. For $t \geq 0$, let G_1, G_2, \dots, G_m be all fuzzy gates aligned at level $t + 1$ and let each fuzzy gate G_k have input variables $\vec{v}_k = (v_{i_1}, v_{i_2}, \dots, v_{i_{a_k}})$ as well as output variables $\vec{w}_k = (w_{j_1}, w_{j_2}, \dots, w_{j_{b_k}})$. Let $\vec{v} = (v_1, v_2, \dots, v_c)$ be the series of all input variables of G_1, G_2, \dots, G_m and let (w_1, w_2, \dots, w_d) be the series of all output variables of them. Now, we define

- $\ell_{\vec{v}'}^{(k)}(\vec{v}_k', \vec{w}_k') = \mu_1(conf_t(v'_1, \dots, v'_c), G_k(\vec{v}_k')(\vec{w}_k'))$, and
- $g(v'_1, \dots, v'_c)(w'_1, \dots, w'_d) = \mu'_1(\{\ell_{\vec{v}'}^{(k)}(\vec{v}_k', \vec{w}_k')\}_{k \in [m]})$,

where, given a variable a , a' means its arbitrary value.

Finally, we define $conf_{t+1}(w'_1, \dots, w'_d)$ to be

$$\mu_3(\{g(v'_1, \dots, v'_c)(w'_1, \dots, w'_d)\}_{(v'_1, \dots, v'_c) \in \text{Supp}(conf_t)}).$$

An *output* (or *outcome*) of C on input s is $conf_t$ when t is the number of levels in C and we write $C(s)$ to denote the outcome of C on s . Moreover, the *size* of C is the total number of fuzzy gates plus the number of all wires between variables and gates used in C .

Now, we consider the aforementioned standard safe tuple of auxiliary operators together with $\mu'_2(\{\alpha_r\}_{r \in B}) = \bigwedge_{r \in B} \alpha_r$.

Theorem 4.1 *Let f be any fuzzy function from $F\Theta_1$ to $F\Theta_2$.*

(1) *If there is a DFTM computing f in polynomial time, then there exists a uniform family of fuzzy circuits of polynomial size that computes f .*

(2) *If there is a uniform family of polynomial-size fuzzy circuits computing f , then there exists a DFTM M that computes f in polynomial time.*

Proof Sketch. We prove only (1). Let $M = (Q, \Delta, \Sigma_1, \Sigma_2, \Gamma, \delta, q_0, F)$ be a DFTM computing f in polynomial time. We want to construct a uniform family $\{C_n\}_{n \in \mathbb{N}}$ of fuzzy circuits that “simulates” M , and thus computes f . For each fuzzy gate g_k at level t , we prepare variables $\{v_{t,\sigma,k}, v'_{t,q,k}\}_{t,\sigma,q,k}$, where $t \in \mathbb{N}$, $q \in \Delta \cup \{\$, \}$, $q \in Q$, and $k \in \mathbb{Z}$, used for inputs and outputs. Intuitively, $v_{t,\sigma,k} = v'_{t,q,k} = 1$, for example, means that M in state q is scanning σ on cell k at time t . We assume a natural and efficient ordering of those variables. It is easy to design a fuzzy gate that “mimics” the behavior of $\delta(q, \sigma_1, \sigma_2)$. Since M halts in time polynomial in the length $\ell(s)$ of s , the size of C_n is also upper-bounded by a certain polynomial in $\ell(s)$. However, g_{k-1}, g_k, g_{k+1} may share the same output variables. To avoid this situation, it is enough to distinguish them by slightly modifying their labels. \square

4.4 Fuzzy Proof Verification Systems

Nondeterminism has played a significant role in the development of the theory of NP-completeness. In the past literature, the notion of polynomial time-bounded nondeterminism has been characterized in various ways. Among them, we are interested in a particular characterization of NP problems, using proof verification processes; namely, NP problems are decision problems that have efficiently verifiable solutions (or proofs). Likewise, we will introduce a model of *fuzzy proof verification system*, in which a verifier tries to check the validity (or correctness) of a given proof, because we believe that this model is more suitable for practical use.

Let us recall the definition of DFTM from Section 4.2 and expand it significantly by adding a mechanism of handling proofs (or certificates). To store such a proof during a verification process, we use an extra read-only infinite tape, called a *proof tape*. A *fuzzy proof verification system* (or an FPVS, in short) is a tuple $\langle N, F\Theta, \Xi, I \rangle$, where $F\Theta$ is a set of all possible fuzzy proofs with $F\Theta \subseteq \mathcal{F}(\Delta_2^*)$ and $N = (Q, \Delta_1, \Delta_2, \Sigma_1, \Sigma_2, \Sigma_3, \Gamma, \delta, q_0, F)$ is a 4-tape DFTM whose first and second tapes are the same as before, whose third tape is a read-only proof tape, and whose fourth one is a write-only output tape. Here, a fuzzy transition function δ is a map from $\delta : (Q - F) \times \Sigma_1 \times \Sigma_2 \times \Sigma_3$ to $\mathcal{F}(Q \times \Sigma_1 \times \Sigma_2 \times \Gamma \times D_1 \times D_2 \times D_3)$, where the i th tape uses a tape alphabet Σ_i and its tape head moves in directions specified as D_i ($= \{0, \pm 1\}$) for each index $i \in [3]$. A configuration is now of the form $upv\ddot{r}\ddot{y}_1hy_2\ddot{w}$, where $y = y_1y_2$ particularly represents the content of the proof tape and y_1hy_2 indicates that its tape head is at the leftmost symbol of y_2 . On input s with proof ϕ , a fuzzy configuration $conf_i$ at time i is defined as follows. Let $conf_0(upv\ddot{r}\ddot{y}_1hy_2\ddot{w})$ denote

$$s(x) \cdot \phi(y) \cdot [upv\ddot{r}\ddot{y}_1hy_2\ddot{w} = q_0\ddot{x}\ddot{y}_0\ddot{h}y\ddot{w}].$$

At time $i + 1$, we assume that $u'pv'\ddot{r}\ddot{y}_1'hy_2'\ddot{w}'$ is of the form $u\tau_1p\tau_2\tau_3\ddot{r}\ddot{y}_1\xi_1h\xi_2\xi_3y_2\ddot{w}\eta$. Let $conf_{i+1}(u'pv'\ddot{r}\ddot{y}_1'hy_2'\ddot{w}')$ denote

$$\mu_2(\{\ell_{d_1,d_2,d_3}(q, \sigma_1, \sigma_2, \sigma_3)\}_{(q,\sigma_1,\sigma_2,\sigma_3),(d_1,d_2,d_3)}),$$

where $(q, \sigma_1, \sigma_2, \sigma_3) \in (Q - F) \times \Sigma_1 \times \Sigma_2 \times \Sigma_3$, $d_1, d_2, d_3 \in \{0, \pm 1\}$, each $\ell_{d_1,d_2,d_3}(q, \sigma_1, \sigma_2, \sigma_3)$ is defined similarly to $\ell_{d_1,d_2}(q, \sigma_1, \sigma_2)$ in Section 4.2; for example, $\ell_{+1,-1,+1}(q, \sigma_1, \sigma_2, \sigma_3)$ equals

$$\mu_1(conf_i(u'\ddot{r}'\ddot{y}'\ddot{w}, \delta(q, \sigma_1, \sigma_2, \sigma_3)(p, \tau_1, \eta, +1, -1, +1)),$$

where $r' = r + 1$ and $u'\ddot{r}'\ddot{y}'\ddot{w}$ expresses the string $uq\sigma_1\tau_2\tau_3v\ddot{r}'\ddot{y}_1h\xi_1\xi_2\xi_3\ddot{w}$.

Let $FIN_N(t, s, \phi)$ be the set of all (uqv, r, y_1hy_2) with $u, v \in \Sigma_1^*$, $y_1y_2 \in \text{Supp}(\phi)$, $q \in F$, $|uv| \leq t$, and $r \in [0, \ell(s)]_{\mathbb{Z}}$. As in Section 4.2, $final_N$ and the running time t of N are similarly defined. Write $N(s, \phi)$ to denote an output (i.e., a fuzzy subset of $\{0, 1\}$) b of N on input s with proof ϕ , where $b(w)$ is defined as

$$\xi(\{final_N(t, uqv\ddot{r}\ddot{y}_1hy_2\ddot{w})\}_{(uqv,r,y_1hy_2) \in FIN_N(t,s,\phi)}).$$

An *outcome* $N(s)$ of the FPVS N on input s is defined as

- $N(s)(1) = \sup_{\phi \in F\Theta} \{N(s, \phi)(1)\}$ and
- $N(s)(0) = \inf_{\phi \in F\Theta} \{N(s, \phi)(0)\}$.

We say that L *admits* N (or N *solves* L) if $N(s) = L(s)$ for all $s \in F\Theta_1$. The notation $\mathcal{L}(N)$ expresses the fuzzy decision problem solved by N .

As a concrete example of fuzzy decision problems admitting FPVS's, we present the Fuzzy Circuit Satisfiability Problem (abbreviated as Fuzzy-Circuit-SAT).

FUZZY-CIRCUIT-SAT (w.r.t. $F\Theta$)

- FUZZY INSTANCE: a (description of) fuzzy circuit C that takes inputs from $F\Theta$.
- FUZZY OUTPUT: output a fuzzy set $b \in \mathcal{F}(\{0, 1\})$, where $b(1) = \sup_{s \in F\Theta} \{C(s)(1)\}$ and $b(0) = \inf_{s \in F\Theta} \{C(s)(0)\}$.

To solve this Fuzzy-Circuit-SAT, it is possible to construct an FPVS using Theorem 4.1. Hence, we obtain the following.

Lemma 4.2 *There exists an FPVS that solves Fuzzy-Circuit-SAT in polynomial time.*

5 Reductions among Fuzzy Problems

The notion of *reducibility* is a basis to NP-completeness and numerous forms of the reducibility have been proposed. Here, we will consider only its simple fuzzification. Firstly, we will define an important fuzzy function class Fuzzy-FPA(γ) of polynomial-time approximately computable fuzzy functions. Using functions in Fuzzy-FPA(γ), we will introduce AF- and APF-reducibilities, which are viewed as natural fuzzifications of Krentel's metric reducibility [9] and AP-reducibility [2].

5.1 Relative Approximation of Membership Degree

In Section 4.2, we have introduced a DFTM model to capture the notion of fuzzy algorithm. However, the exact use of DFTM's seems too restrictive to solve a wide range of practical fuzzy problems. It is thus desirable to allow the DFTM's to "approximate" the outcomes of the fuzzy problems.

To describe this "approximation," let $\gamma : \mathbb{N} \rightarrow [1, \infty)$ be any crisp function, called an *imprecision tolerance parameter*. Let $F\Theta \subseteq F\Delta^*$ for alphabet Δ as before. For any two fuzzy subsets F and G of the universe $F\Theta$, we say that F is a γ -approximation of G if $F(x)/\gamma(|x|) \leq G(x) \leq \gamma(|x|)F(x)$ for any element x in $F\Theta$.

Definition 5.1 (approximate solving) Let M be either an DFTM or an FPVS. We say that M γ -approximately solves fuzzy problem L if, for any admissible fuzzy input s , M takes s as an input and produces a certain fuzzy solution $M(s)$ that is a γ -approximation of $L(s)$.

In ordinary computational complexity theory, FP denotes the set of all crisp functions on crisp strings computable in polynomial time. We introduce an analogous "fuzzy" function class denoted by Fuzzy-FPA(γ) using the γ -approximability.

Definition 5.2 (Fuzzy-FPA(γ)) Let $\gamma : \mathbb{N} \rightarrow [1, \infty)$ be any imprecision tolerance parameter and fix a safe tuple Ξ . For convenience, we define Fuzzy-FPA $_{\Xi}(\gamma)$ to be the set of all (combinatorial generic) fuzzy problems, each of which can be γ -approximately solved by a certain polynomial-time DFTM. The "PA" stands for "polynomial-time approximate." When Ξ is clear from the context, we drop Ξ and write Fuzzy-FPA(γ).

Let us introduce four classes of fuzzy functions. The notation *const* expresses the set of all constant functions (which we usually identify with "constants"), *poly* does the set of all polynomials, and *exp* does the set of all exponential functions. We then obtain the following chain of containments: Fuzzy-FPA(1) \subseteq Fuzzy-FPA(*const*) \subseteq Fuzzy-FPA(*poly*) \subseteq Fuzzy-FPA(*exp*).

We can show that FP is a crisp part of Fuzzy-FPA if we take the following method of fuzzifying crisp objects. For each crisp string x in Δ^* , the notation \hat{x} denotes its specific fuzzified object $\{(x, 1)\} \cup \{(y, 0) \mid y \in \Delta^* - \{x\}\}$ in $\mathcal{F}(\Delta^*)$.

Lemma 5.3 For any crisp function $f : \Delta^* \rightarrow \Gamma^*$ in FP, there exists a fuzzy function $g \in \text{Fuzzy-FPA}(1)$ such that, for every $(x, y) \in \Delta^* \times \Gamma^*$, $f(x) = y$ if and only if $g(\hat{x}) = \hat{y}$.

Lemma 5.4 Let $\mathcal{F} \in \{1, \text{const}, \text{poly}, \text{exp}\}$. Fuzzy-FPA(\mathcal{F}) is closed under functional composition; namely, for any two elements $f, g \in \text{Fuzzy-FPA}(\mathcal{F})$, the function h defined as $h(s) = g(f(s))$ for every s is also in Fuzzy-FPA(\mathcal{F}).

5.2 Approximate Fuzzy Reductions

To compare the computational complexities of two (generic) fuzzy problems, we need to devise a notion of "fuzzy reducibility," which is a mechanism (similar to the metric reducibility in [9]) of transforming instances of one decision problem to instances of another decision problem so that corresponding outputs of those instances are close enough.

Definition 5.5 (AF-reducibility) Let F be any fuzzy function mapping $F\Theta_1$ to $F\Theta_2$ and let G be any fuzzy function from $F\Theta_3$ to $F\Theta_4$. We say that F is *polynomial-time γ -approximately fuzzy reducible* (or *AF(γ)-reducible*, in short) to G if there exist two functions f, g in Fuzzy-FPA(\mathcal{F}) with $f : F\Theta_1 \rightarrow F\Theta_3$ and $g : F\Theta_1 \times F\Theta_4 \rightarrow F\Theta_2$ and a function $\gamma \in \mathcal{F}$ such that, for any $s \in F\Theta_1$, $g(s, G(f(s)))$ is a γ -approximation of $F(s)$. In this case, we write $F \leq_{\text{AF}}^{(\mathcal{F})} G$. This triplet (f, g, γ) is called an *AF(\mathcal{F})-reduction* of F to G .

The next lemma establishes the reflexivity and the transitivity of the AF-reducibility. Thus, the AF(\mathcal{F})-reducibility forms a partial order in the set of all fuzzy problems.

Lemma 5.6 Let $\mathcal{F} \in \{1, \text{const}, \text{poly}, \text{exp}\}$. For any three fuzzy problems A, B, C having the same range, it holds that (1) $A \leq_{\text{AF}}^{(\mathcal{F})} A$ and (2) $A \leq_{\text{AF}}^{(\mathcal{F})} B$ and $B \leq_{\text{AF}}^{(\mathcal{F})} C$ imply $A \leq_{\text{AF}}^{(\mathcal{F})} C$.

Reducibility between two fuzzy optimization problems, in contrast, requires a more delicate treatment. As a natural fuzzification of the ordinary AP-reducibility (see, e.g., [2]), we introduce the following APF-

reducibility. Let $R(s, t) = \max\{|m(s, u)/m^*(s)|, |m^*(s)/m(s, u)|\}$ for a given fuzzy measure function m . We set $\mathbb{Q}^{>1} = \{r \in \mathbb{Q} \mid r > 1\}$.

Definition 5.7 (APF-reducibility) For any two fuzzy optimization problems $A = (I_1, SOL_1, m_1, goal)$ and $B = (I_2, SOL_2, m_2, goal)$, we say that A is *polynomial-time approximation-preserving fuzzy (APF) reducible to B* with respect to \mathcal{F} , denoted by $A \leq_{\text{APF}}^{(\mathcal{F})} B$, if the following condition holds: there exist a pair (f, g) of fuzzy functions and a constant $c > 0$ such that

- (1) $f(s, r) \in I_2$ for any $s \in I_1$ and $r \in \mathbb{Q}^{>1}$;
- (2) $SOL_1(s) \neq \emptyset$ implies $SOL_2(f(s, r)) \neq \emptyset$;
- (3) $u \in SOL_2(f(s, r))$ implies $g(s, u, r) \in SOL_1(s)$;
- (4) $f, g \in \text{Fuzzy-FPA}(\mathcal{F})$ for each fixed $r \in \mathbb{Q}^{>1}$;
- (5) $R_2(f(s, r), u) \leq r \rightarrow R_1(s, g(s, u, r)) \leq 1 + c(r - 1)$.

6 Fuzzy Decision Problems

In ordinary complexity theory, decision problems are identified with sets of words (or strings) and they are also called languages. The fundamental complexity classes of languages are P and NP. Analogously, we recognize two special classes of fuzzy decision problems.

Definition 6.1 (Fuzzy-PA(γ), Fuzzy-NPA(γ)) Let Fuzzy-PA(γ) be a subclass of Fuzzy-FPA(γ), which consists only of fuzzy decision problems. Moreover, Fuzzy-NPA(γ) denotes the set of all fuzzy decision problems that can be γ -approximately solved by polynomial-time FPVS's.

Since DFTMs are a special case of FPVS's, it immediately follows that $\text{Fuzzy-PA}(\gamma) \subseteq \text{Fuzzy-NPA}(\gamma)$ for any γ . As in Section 5.1, we write Fuzzy-PA (resp., Fuzzy-NPA) for Fuzzy-PA(const) (resp., Fuzzy-NPA(const)).

Notice that DFTMs are in fact an extension of ordinary NTMs. Therefore, it is not surprising to show that all NP sets can be solved by DFTMs in polynomial time if we use an appropriate fuzzification (i.e., if we assign appropriately-chosen possibility degrees). Recall from Section 5.1 the fuzzification \hat{x} of crisp string x . If a fuzzification \hat{L} of crisp language L is defined to satisfy that $\hat{L}(\hat{x}) = \{(0, 0), (1, 1)\}$ if $x \in L$ and $\hat{L}(\hat{x}) = \{(0, 1), (1, 0)\}$ otherwise, we can show that, for any language $L \in \text{NP}$, \hat{L} belongs to Fuzzy-PA(1).

Proposition 6.2 For every set $A \in \text{NP}$ over alphabet Δ , there exists a fuzzy decision problem $B \in \text{Fuzzy-PA}(1)$ such that $A = \{x \in \Delta^* \mid B(\hat{x})(1) = 1\}$ and $\bar{A} = \{x \in \Delta^* \mid B(\hat{x})(1) = 0\}$.

Hereafter, we assume that $\mathcal{F} \in \{1, \text{const}, \text{poly}, \text{exp}\}$.

Lemma 6.3 The fuzzy complexity class Fuzzy-PA(\mathcal{F}) is closed under AF(\mathcal{F})-reductions.

Proof Sketch. It suffices to show that, for any two fuzzy problems A and B , if A is AP-reducible to B and B is in Fuzzy-PA, then A is also in Fuzzy-PA(\mathcal{F}). Since $A \leq_{\text{AF}}^{(\mathcal{F})} B$, take a reduction (f, g, γ) with $f, g \in \text{Fuzzy-FPA}(\mathcal{F})$ that reduces A to B . Since $B \in \text{Fuzzy-PA}(\mathcal{F})$, take a DFTM M that solves B . It suffices to consider the following DFTM G : on input s , generate all $x \in \text{Supp}(s)$ in parallel, compute M_f on them, starts the simulation of N , and compute M_g . \square

Now, we introduce a key concept of “completeness.”

Definition 6.4 (completeness) We say that a fuzzy decision problem A is *complete* for Fuzzy-NPA(\mathcal{F}) (or simply, Fuzzy-NPA(\mathcal{F})-complete) if (i) A is in Fuzzy-NPA(\mathcal{F}) and (ii) for every fuzzy decision problem B in Fuzzy-NPA(\mathcal{F}), A is AF(\mathcal{F})-reducible to B .

Lemma 6.5 Let A be a fuzzy decision problem complete for Fuzzy-NPA(\mathcal{F}). If A is in Fuzzy-PA(\mathcal{F}), then Fuzzy-PA(\mathcal{F}) = Fuzzy-NPA(\mathcal{F}) holds.

Proof Sketch. The containment Fuzzy-PA \subseteq Fuzzy-NPA is obvious. Next, we will show the other containment. Take any fuzzy problem B in Fuzzy-NPA. By the definition of “completeness,” B is AP-reducible to A . Assume that A is in Fuzzy-PA. By Lemma 6.3, it follows that B is also in Fuzzy-PA. Thus,

Fuzzy-NPA \subseteq Fuzzy-PA. □

Finally, we will demonstrate the existence of complete problems for Fuzzy-NPA(1). Our choice of such problem is Fuzzy-Circuit-SAT, defined in Section 4.4.

Theorem 6.6 *The fuzzy problem Fuzzy-Circuit-SAT is complete for Fuzzy-NPA(1).*

Proof Sketch. Recall from Lemma 4.2 that *Fuzzy-Circuit-SAT* is in Fuzzy-NPA(1). What remains to show is that any problem, say, A in Fuzzy-NPA(1) is AF-reducible to *Fuzzy-Circuit-SAT*. Given such A , take an FPVS N for A . By Theorem 4.1(1), we can choose a family of fuzzy circuits that “simulates” N when input s and proof ϕ are initially given. It thus suffices to define an AF-reduction pair (f, g) as follows. Let $f(s)$ be a circuit obtained from C by incorporating it with s (treating ϕ as only a true input). The function g is defined as the identity function. □

7 Fuzzy Optimization Problems

Briefly, we will discuss the computational complexity of fuzzy optimization problems. The theory of NP optimization problems has made a huge success in classifying “complete” problems in NPO (the class of NP optimization problems). We will pay our attention to “fuzzy NPA” optimization problems.

Definition 7.1 (Fuzzy-NPO(γ)) A fuzzy NPA(γ) optimization problem (or a fuzzy NPAO(γ) problem) P is a fuzzy optimization problem $(I, SOL, m, goal)$ as defined in Section 3 such that $I \circ SOL$ is in Fuzzy-PA(γ) and m is in Fuzzy-FPA(γ). We write Fuzzy-NPAO(γ) for the class of all fuzzy NPAO(γ) problems.

Similar to Fuzzy-NPA(1), Fuzzy-NPAO(1) has complete problems under AFP(1)-reductions.

Theorem 7.2 *There exists a fuzzy optimization problem that is complete for Fuzzy-NPAO(1) under APF(1)-reductions.*

References

- [1] M. F. Abbod, D. G. von Keyserlingk, D. A. Linkens, and M. Mahfouf. Survey of utilisation of fuzzy technology in medicine and healthcare. *Fuzzy Sets and Systems* 120, 331–349, 2001.
- [2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, 2003.
- [3] M. Blue, B. Bush, and J. Puckett. Unified approach to fuzzy graph problems. *Fuzzy Sets and Systems*, 125, 355–368, 2002.
- [4] M. Doostfatemeleh and S. Kremer. New directions in fuzzy automata. *Int. J. Approx. Reason.*, 38, 175–214, 2005.
- [5] M. R. Garey and D. S. Johnson. *Computers and Intractability: Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [6] R. Goetschel Jr. and W. Voxman. Fuzzy circuits. *Fuzzy Sets and Systems*, 32, 35–43, 1989.
- [7] M. Hanss. *Applied Fuzzy Arithmetic: An Introduction with Engineering Applications*. Springer, 2010.
- [8] K. Hirota. Fundamentals of fuzzy logical circuits. In *Proc. of IJCAI '91 Workshops on Fuzzy Logic and Fuzzy Control*, Lecture Notes in Computer Science, vol. 833, pp. 143–157, 1994.
- [9] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst.*, 36, 490–509, 1988.
- [10] E. T. Lee and L. A. Zadeh. Note on fuzzy languages. *Inform. Sci.*, 4, 421–434, 1969.
- [11] L. Li, S. N. Kabadi, and K. P. K. Nair. Fuzzy models for single-period inventory problem. *Fuzzy Sets and Systems* 132, 273–289, 2002.
- [12] E. S. Santos. Fuzzy algorithms. *Inform. Control*, 17, 326–339, 1970.
- [13] L. A. Zadeh. Fuzzy sets. *Inform. Control* 8, 338–353, 1965.
- [14] L. A. Zadeh. Fuzzy algorithms. *Inform. Control*, 12, 94–102, 1968.